Contents

1 General description

This document provides a detailed description of the calibration methodology and examples of the calibration procedure on the chip. It shows how to calibrate both temperature and pressure sensors in one measurement sequence, resulting in 14 calibration parameters. These 14 coefficients or parameters (2 for T and 12 for P) are stored in a One Time Programmable (OTP) memory.

In order to perform a pressure and temperature calibration several steps need to be conducted:

- Measuring raw capacitance and temperature output values together with their associated pressure and temperature reference values.
- Linearization of the measured C,P,T points by polynomial curve fitting and extraction of the calibration coefficients.
- Calculation of calibration parameter values to be written into the OTP memory of the chip taking into account the preset ROM offset values. The final 2s complement correction values are distributed over the P2RAM and need to be entered as hexadecimal codes into 28 different addresses.
- Actual storing these values in the OTP memory chip using a burning sequence. The hardware and set-up needed for OTP fusing are mentioned in Paragraphs 5.3 and 5.4.

2 Functional Description

The chip integrates an absolute pressure sensor and a temperature sensor. Since the pressure sensor is cross sensitive for temperature, every P readout relies on the most recent T measurement. Measurement performance, power consumption, and device behaviour can be configured to fulfil the requirements of different use cases. Features include configurable conversion time, configurable oversampling, interrupts, and a 32-slot memory that can be used as FIFO or as a moving average filter.

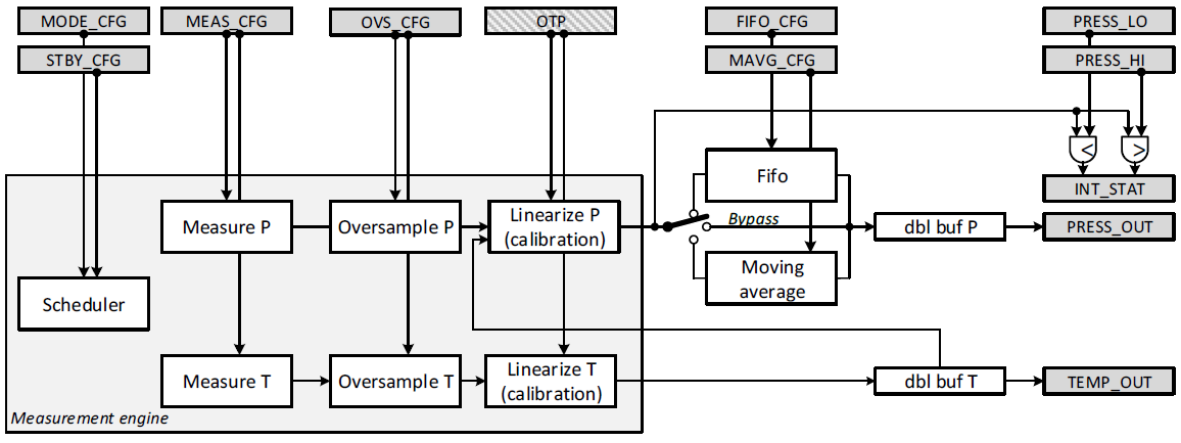

Figure 1 Pressure sensor data flow model

Each raw capacitance and temperature value are linearized using a polynomial correction formula. The calibration parameters are stored in OTP i.e. One Time Programmable Memory (see Figure 1).

2.1 Temperature Sensor

The temperature sensor measures the junction temperature and outputs a calibrated value proportional to the absolute temperature. The junction temperature is measured using a high-precision ADC, as shown in Figure 2.

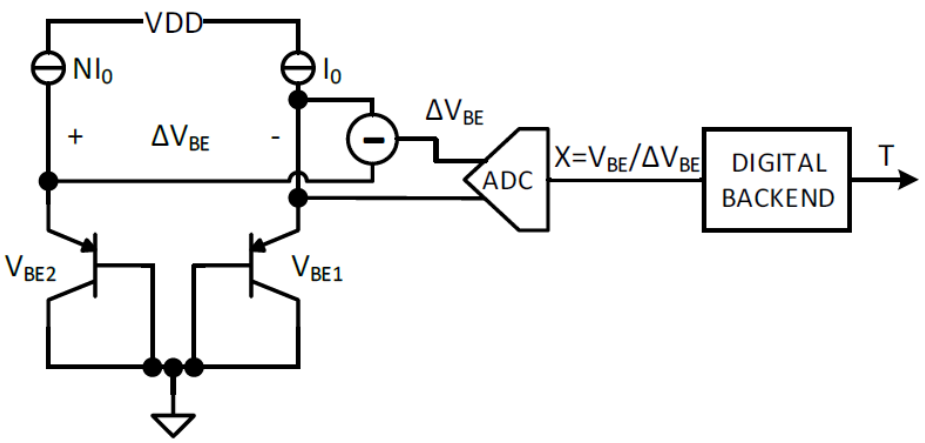

Figure 2: Temperature sensor structure

Temperature can be calculated as follows:

T°C=TEMP_OUT/128 – 273.15

T°K=TEMP_OUT/128

2.2 Absolute Pressure Sensor

The capacitive pressure sensor determines the ambient absolute pressure and outputs a calibrated and linearized value. Referring to the block diagram in **Error! Reference source not found.**, the transducer CX consists of a large-area membrane that acts as a capacitor. The capacitance change is proportional to the change in pressure. The capacitance is measured by a high-precision 3rd order sigma-delta converter.

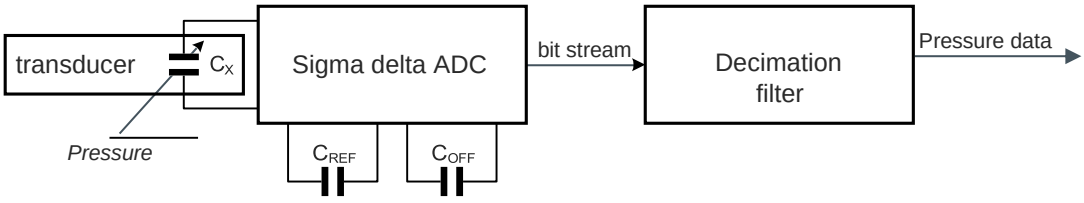

Figure 3 Absolute Pressure Sensor

The raw capacitance reading is automatically converted to an absolute pressure value through an optimized algorithm that compensates for the non-linearity of the transducer and for the temperature dependency, by combining calibration data stored in the internal OTP and the latest temperature measurement. Absolute pressure is calculated as follows:

$$P_{Pa}=PRESS\_OUT/64$$

NOTE: The PRESS_OUT value is already calibrated, linearized and temperature-compensated; it requires a division by 64 to achieve a pressure reading in Pa.

2.3 Characteristics of the capacitive pressure sensor

The capacitive pressure sensor membrane that is integrated on the ASIC exhibits the typical non-linear capacitance-pressure behaviour. The non-linear C-P behaviour can be understood because the capacitance increases, to a first order approximation, with the inverse distance between the electrode plates (C ~ 1 / gap). In Figure 4 the capacitance pressure curve is depicted for several membrane sizes showing the non-linear C-P dependence before touchdown of the top electrode and a fairly C-P linear behaviour in the “collapsed state”.

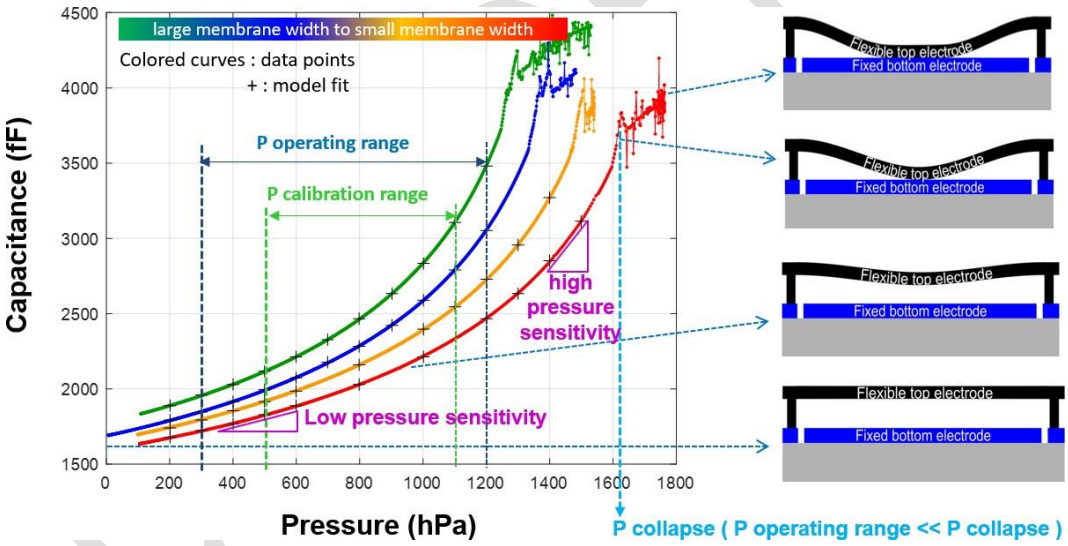

Figure 3: Capacitance - pressure curves for different membrane widths

The lowest pressure sensitivity is obtained at low pressure when the membrane deflection is the lowest and the membrane to bottom electrode separation is maximum. The highest pressure sensitivity is achieved just before the membrane touches the bottom of the cavity. The touch point (collapse) pressure is designed to be higher than the maximum pressure operating range in order to get correct results out of the linearization. Please note that the device should not be operated in collapsed mode because the linearization algorithm is only applicable to pressures before collapse.

Capacitive pressure sensors typically require a large number of calibration points and non-linear curve fitting routines to get an accurate pressure output value. Calibration methods are based on higher order polynomial fits on C-P data points. The more calibration data points are used the better the accuracy.

In order to come to a pressure output value that is compensated for temperature, the sensor capacitance is internally linearized with an embedded polynomial correction function that takes the measured temperature into account as well. In this document the procedure is described to derive the polynomial constants and how to write these constants into the OTP of the ASIC.

3 General description of the calibration flow

During the calibration procedure, the raw capacitance output and the raw temperature output are recorded and logged from the ASIC. Depending on the required accuracy, a certain number of pressure and temperature points are required to perform the calibration of the pressure sensor and temperature sensor. The logging of the raw capacitance and temperature values together with the pressure readings from the reference pressure sensor and the temperature readings of the reference temperature sensor must be performed on an external device/computer. A two dimensional polynomial least squares fitting routine will be applied to the recorded capacitance, pressure and temperature points to derive the calibration constants that should be written in the OTP memory. Please note that only the calibration constants will be stored in the OTP and not the calibration points.

In order to achieve the required accuracy of both the pressure and temperature outputs, sufficient stabilization time must be applied after setting new pressure and temperature set points. During the measurement of the transducer both the pressure and temperature variation must be minimized and kept within narrow limits.

The set points of the pressure and temperature can be optimized depending on the P and T range that should be covered. Outside the outer P and T calibration points the calibration error becomes typically worse than in between. However, a very broad separation between the pressure and temperature points will induce a larger error in between these points.
Please note that the temperature sensor does not have to be accurately calibrated to correct for the temperature sensitivity of the pressure sensor. The linearity of the temperature sensor is sufficiently good to avoid introducing large errors in the pressure compensation algorithm. Using the factory predefined constants the temperature output can be +/-2°C off depending on the temperature.

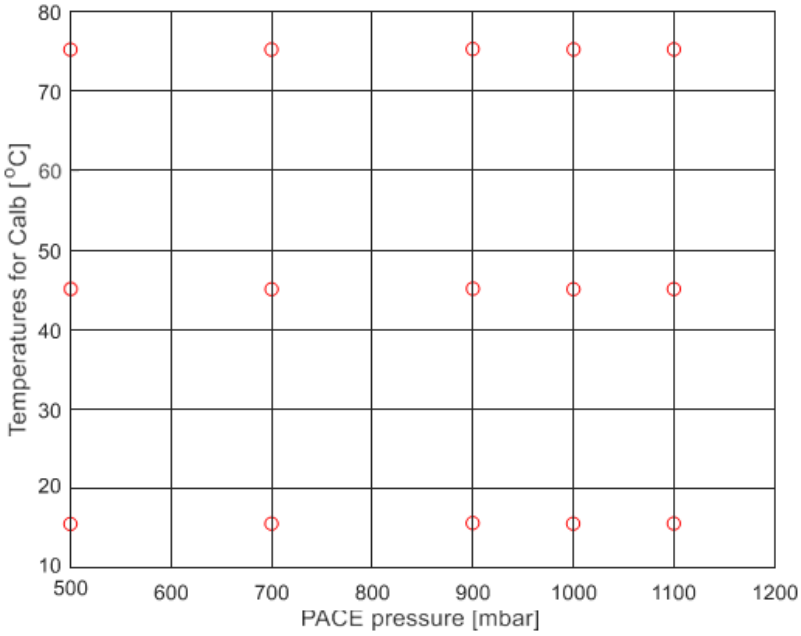

Figure 5 Example of pressure & temperature calibration set points

The pressure sensor membrane is temperature sensitive due to CTE mismatch of the membrane and the underlying CMOS die. Therefore, the pressure output needs to be compensated for temperature variations. In addition, the CTE mismatch between the PS die and the package or laminate could introduce additional stress in the pressure sensor die. If not properly mitigated the latter could lead to significant first and second order temperature dependencies. In order to determine these effects and to mitigate them, it is advised to use three temperatures for the pressure sensor calibration. These temperatures are depending on the required temperature operating range and the non-linearity of the C (P, T) curve. A typical range of temperatures for the three temperature set points are -5 °C … 15 °C, 25 °C … 45 °C and 55 °C … 75 °C.

It is advised to apply at least five pressure points in order to achieve good pressure calibration results (<0.5 hPa accuracy): for a pressure range from 500 hPa to 1100 hPa one could apply e.g. 1100 hPa, 1000 hPa, 900 hPa, 700 hPa, and 500 hPa. The order of points is only of importance with respect to the time required to reach the set point and a stable reading. Please note that outside the proposed pressure range i.e. from 300-500 hPa and from 1100 to 1200 hPa an error is expected larger than 1 hPa.

4 Calibration procedure

4.1 Flow chart summary

The procedure of performing a combined T and P calibration is illustrated and summarized in the following flowchart shown in Figure 6.

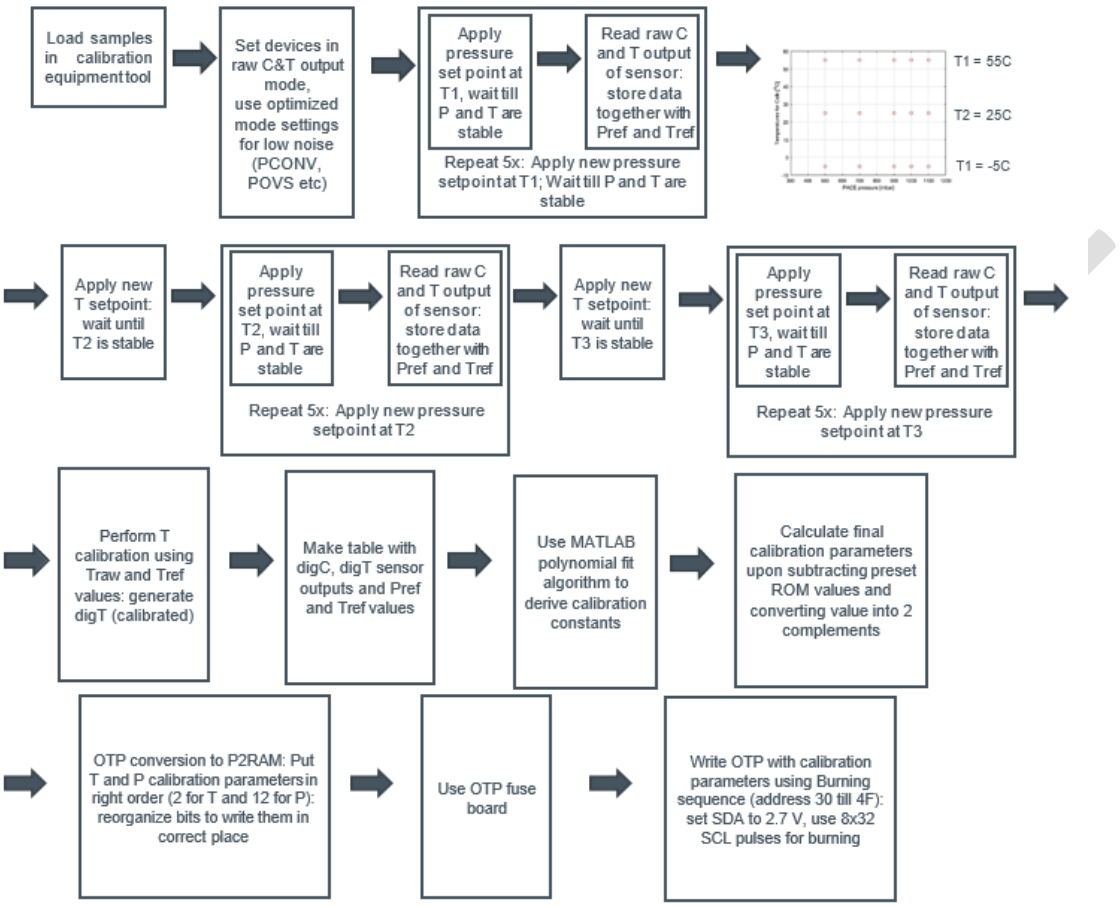

Figure 6: Flowchart for the chip P&T calibration

4.2 Calibration data collection

4.2.1 Measurement procedure

Below an example is given for a possible approach to perform the calibration procedure:

- **Set temperature of test cell to a predefined first T set point**: select temperature in a range from 25°C … 45°C (e.g. 45 °C)
- Load packages into sockets of test cell of pressure test chamber
- Close pressure chamber lid
- **Set pressure controller to first pressure set point**: e.g. choose a P set point close to ambient pressure in order to save stabilization time e.g. choose for first pressure set point 1100 hPa.

- Wait for pressure and temperature to be constant e.g. within +/-10 Pa limit and 0.1 ºC. Depending on the system and the required accuracy, a fixed stabilization time can also be used.

- During the stabilization electrical parameters of the chip could be measured.

- Measure UID of each device that is loaded to check if good contact is being made.

- Measure *raw i.e. uncalibrated temperature* output.
    - Perform T measurement of calibrated reference temperature sensor: log reference temperature.
    - Perform T measurement of DUT: Use T2D read-out of ASIC, log raw temperature value
- Measure *raw i.e. uncalibrated capacitance* output at constant T
    - Perform P measurement of calibrated reference pressure sensor
    - Perform C measurement of DUT, use C2D read-out of ASIC, log raw capacitance output

- **Repeat this for second, third, fourth and fifth pressure point:**
    - Pressure point i.e. 1000 hPa, 900 hPa, 700 hPa, 500 hPa
    - Wait for pressure to be constant within +/-10 Pa limit (or use fixed e.g. 6 s stabilization time)
    - Measure T on reference T sensor and DUT
    - Measure P on reference pressure sensor and C raw on DUT
    - Store all C-P-T points together with the UID on a computer

- **Heat the system to a second temperature or use the 2nd chamber at elevated temperature**
    - Select temperature in T range of 55 ºC … 75 ºC: e.g. 75 ºC

- **Repeat sequence of measuring five pressure points, now at a different temperature**
    - Use similar pressure points as at first temperature (e.g. 1100 hPa, 1000 hPa, 900 hPa, 700 hPa, and 500 hPa)
    - Store C-P-T points with UID on computer

- **Heat the system to a third temperature or use a 3rd chamber at low temperature.**
    - Select temperature in T range of -5 ºC … 15 ºC: e.g. 15 ºC

- **Repeat the sequence of measuring five pressure points for the third time**
    - Use similar pressure points as at first temperature (e.g. 1100 hPa, 1000 hPa, 900 hPa, 700 hPa and 500 hPa)
    - Store all C-P-T points together with the UID on a computer.

4.2.2 Chip configuration settings

	Settings							Registers			Registers			
	Pconv time (ms)	P_OVS	T_OVS	Time stby (ms) (0=cont, 1=single)	PT rate	T noise (mK rms)	P noise @100kPa (Pa rms)	MEAS_CFG (hex)	STBY_CFG (hex)	OVS_CFG (hex)	MEAS_CFG (bin)	STBY_CFG (bin)	OVS_CFG (bin)	total conversion time (ms)
4ms+1ms	1	1	1	1	1	22.0	0.90	00	01	00	0	1	0	9
	1	2	1	1	1	22.0	0.73	00	01	08	0	1	1000	10
	1	4	2	1	1	16.1	0.53	00	01	11	0	1	10001	13
	1	8	4	1	1	11.8	0.39	00	01	1A	0	1	11010	19
	1	16	8	1	1	8.6	0.29	00	01	23	0	1	100011	31
	1	32	16	1	1	6.3	0.21	00	01	2C	0	1	101100	55
	1	64	32	1	1	4.6	0.15	00	01	35	0	1	110101	103
	1	128	64	1	1	3.4	0.11	00	01	3E	0	1	111110	199
8ms+2ms	2	1	1	1	1	22.0	0.71	08	01	00	1000	1	0	13
	2	2	1	1	1	22.0	0.60	08	01	08	1000	1	1000	15
	2	4	2	1	1	16.1	0.44	08	01	11	1000	1	10001	20
	2	8	4	1	1	11.8	0.32	08	01	1A	1000	1	11010	30
	2	16	8	1	1	8.6	0.24	08	01	23	1000	1	100011	50
	2	32	16	1	1	6.3	0.17	08	01	2C	1000	1	101100	90
	2	64	32	1	1	4.6	0.13	08	01	35	1000	1	110101	170
	2	128	64	1	1	3.4	0.09	08	01	3E	1000	1	111110	330
16ms+4ms	4	1	1	1	1	22.0	0.59	10	01	00	10000	1	0	21
	4	2	2	1	1	16.1	0.43	10	01	09	10000	1	1001	26
	4	4	4	1	1	11.8	0.32	10	01	12	10000	1	10010	36
	4	8	8	1	1	8.6	0.23	10	01	1B	10000	1	11011	56
	4	16	16	1	1	6.3	0.17	10	01	24	10000	1	100100	96
	4	32	32	1	1	4.6	0.12	10	01	2D	10000	1	101101	176
	4	64	64	1	1	3.4	0.09	10	01	36	10000	1	110110	336
	4	128	128	1	1	2.5	0.07	10	01	3F	10000	1	111111	656

Table 1: Advised configuration modes and register settings

It is advised to choose a setting with sufficient low noise (~0.3 Pa rms) and not too long conversion times (< 50ms): e.g. select : MEAS_CFG “08” (8ms+2ms conversion time), STBY_CFG “01” (single shot mode), OVS_CFG “1A” (P oversampling 8x, T oversampling 4x, PT rate =1).
Table 8 provides an overview of possible modes and register settings. Please note that the value 0x01 must be written to register 0x56 to set the device in raw output mode.

As an example, a series of commands/instructions is listed below.

write 0x08 to 0x06 // MODE_GFG: soft reset
write 0x80 to 0x06 // MODE_GFG: power-up
wait 0.5ms // wait for power-up
write 0x67 to 0x7F // enable additional configurations
write 0x01 to 0x56 // select raw capacitance output
write 0x08 to 0x07 // MEAS_CFG: 2ms conversion time, PT rate=1
write 0x01 to 0x08 // STBY_CFG: one shot mode
write 0x1A to 0x09 // OVS_CFG: P oversampling=8, T oversampling=4
write 0x13 to 0x06 // MODE_GFG: start measurement without FIFO
wait 35ms // Conversion time P->8+7*2 T->4+3 +margin
read 3 bytes from 0x17 // PRESS_OUT: read capacitance value
read 2 bytes from 0x1A // TEMP_OUT: read temperature value

4.2.3 Example data from calibration measurement

After taking all measurements, the collected data can be listed as in Table 2 below. At three temperatures and five pressures, the device capacitance is recorded together with reference values for T and P.

In this example the temperatures are set to 60, 40 and 20 °C. The pressure reference is recorded by a PACE device. The digital capacitance output of the device is in column **"RAW_C"**. The temperature reference device is a T12 temperature probe, of which the readings are in column **"Tprobe"**.

Reading temperature from the chip results in a digital value (**"UNCAL tempCodes"**) which can be divided by 128 to reach temperature in Kelvin. After subtraction of 273.15, one obtains the values in Celcius listed in column **"Tens"**.

T_idx	P_idx	UID	setT	setP	PACE ref	Tprobe	RAW_C	UNCAL temp-Codes	Tens
1	1	0x21c0f135	60	1000	1000.068	59.93	4141951	42144	56.10
1	3	0x21c0f135	60	1100	1099.875	59.96	4498850	42148	56.13
1	5	0x21c0f135	60	800	800.084	59.77	3604105	42134	56.02
1	7	0x21c0f135	60	500	500.113	59.56	3053623	42110	55.83
1	9	0x21c0f135	60	900	900.016	59.83	3849013	42120	55.91
2	1	0x21c0f135	40	1000	1000.013	39.92	4131078	39600	36.23
2	3	0x21c0f135	40	1100	1099.934	39.96	4483630	39604	36.26
2	5	0x21c0f135	40	800	800.025	39.75	3599036	39591	36.15
2	7	0x21c0f135	40	500	500.086	39.52	3052867	39564	35.94
2	9	0x21c0f135	40	900	900.029	39.80	3841729	39573	36.01
3	1	0x21c0f135	20	1000	999.796	19.95	4118067	37058	16.37
3	3	0x21c0f135	20	1100	1099.879	19.99	4465710	37060	16.38
3	5	0x21c0f135	20	800	800.026	19.76	3593123	37045	16.26
3	7	0x21c0f135	20	500	500.131	19.54	3051817	37017	16.05
3	9	0x21c0f135	20	900	900.065	19.84	3833226	37032	16.16

Table 2: Examples of calibration data acquisition

4.3 Temperature calibration

We will use the temperature measurements at pressure set point P=1000 hPa. By excluding all other pressure set points, we make sure that the small thermal effect of raising and lowering pressure is not influencing the calibration of the temperature sensor. The set point P=1000 hPa is maintained while stabilizing the setup to a new temperature set point for enough time to reach thermal equilibrium.

From the Table 2 we select the relevant lines for temperature calibration: In Table 3 the results are shown.

setT	setP	PACE ref	Tens (0,0)	Tdig0,0	Tprobe	Tens (A, α)
60	1000	1000.068	56.10	42144	59.93	59.92
40	1000	1000.013	36.23	39600	39.92	39.93
20	1000	999.796	16.37	37058	19.95	19.95

Table 3: Results of T calibration

The last column **Tens (A,α)** is the result of the calibration, which is described below. It should be noted that **"Tens"** is the temperature from the device. Before calibration, it has two zeros (0,0) in the calibration registers. In that state it will provide a temperature close to the real temperature, but with an offset of a few degrees.

The last column is what the device would return when the two calibration parameters are stored in the OTP memory. In this case the numbers are A=904, α=161. The following section shows how these numbers are determined.
The temperatures before calibration Tens(0,0) are represented in digital form (column Tdig0,0) by taking:

$$Tdig0,0 = 128 * [Tens(0,0) + 273.15]$$

The internal raw values X for temperature are then (using alpha_trim = 0 and A_trim = 0) calculated step-by-step: (numerical values are indicated per line for reference)

temp_out (is Tdig0,0 in Table)	37058	39600	42144
alpha = alpha_trim * 512 + 8388608;	8388608		
A = A_trim * 8192;	0		
X4 = ((temp_out)+2029)*4;	156348	166516	176692
X3 = (X4 ./ (1 + A / 2^30));	156348	166516	176692
X31 = (X3 ./ (1 + 23068672 /2^27));	133417	142094	150777
X2=X31*(2^25)/alpha;	533668	568375	603109
X1 = (2^43 ./ X2);	16482337	15475872	14584590
X = ((X1 - alpha) / 32);	252929	221477	193624

These raw values (252929, 221477, 193624) are internal values that the temperature sensor will measure, irrespective of what values of A and alpha are set in the OTP. Calibration now comes down to finding the A and alpha values that make from these raw values X the proper output temperatures T. These should be the same as the reference temperatures recorded in the measurement.

We have implemented an iterative search algorithm in MATLAB to obtain these optimal values. See in the Appendix the functions “newfinding”, “findvals”, “Temp2Raw” and “Raw2Temp”. The optimization is started by calling the function newfinding with:

```
[alphanew,anew]=newfinding(Tens(0,0), Tprobe);
```

For example:

```
[alphanew,anew]=newfinding([56.1; 36.23; 16.37], [59.93; 39.92; 19.95]);
```

Here we have inserted the columns of the table above. The resulting values are A=904, alpha=161. These two values minimize the least-squares error between what the chip T-sensor would output and the T-reference. In principle, only two measurements are needed to find two parameters. In this example we have used three temperatures. There is no limit on the number of temperatures to insert in this optimization.

4.4 Pressure calibration

For pressure calibration, a multi-dimensional polynomial function is fitted to the 15 measured P-C-T points. This results in 12 pressure calibration coefficients (b40 to b00), which are subsequently written to the OTP (see also section 5). In the following, the example (with data from Table 2) from the previous sections is continued.

4.4.1 Calculate the calibrated temperature values

CalibratedT	P_ref	digT	digC
59.92	1000.068	42633	4141951
59.95	1099.875	42637	4498850
59.84	800.084	42623	3604105
59.66	500.113	42599	3053623
59.73	900.016	42609	3849013
39.93	1000.013	40074	4131078
39.97	1099.934	40078	4483630
39.86	800.025	40065	3599036
39.65	500.086	40038	3052867
39.72	900.029	40047	3841729
19.95	999.796	37516	4118067
19.96	1099.879	37518	4465710
19.85	800.026	37503	3593123
19.63	500.131	37475	3051817
19.74	900.065	37490	3833226

Table 4: Input values for P calibration

The polynomial will fit to the calibrated temperature values (as A and alpha are already written to the memory of the device). This results in the values depicted in Table 4 above.

4.4.2 Raw value conversion

The data is further processed prior to fitting. For the (internally scaled) polynomial calculation we find the values for capacitance, temperature and pressure via:

```
C_poly = (-4682800+((2^43)./((digC-349526)./2)))/(2^21);
T_poly = (digT-30145)/(2^12);
P_poly = ((100*P_ref)-75000)/(2^16);
```

The resulting C-poly, T_poly, and P_poly values that will be used as internal input values for the polynomial fit is shown in Table 5 below:

C_poly	T_poly	P_poly
-0.021	3.04883	0.38157
-0.21125	3.0498	0.53387
0.34455	3.04639	0.07642
0.86925	3.04053	-0.3813
0.16416	3.04297	0.22891
-0.01464	2.42407	0.38149
0.34857	2.42188	0.07633
0.87012	2.41528	-0.38134
0.16916	2.41748	0.22893
-0.00698	1.79956	0.38116
-0.19498	1.80005	0.53387
0.35327	1.79639	0.07633
0.87132	1.78955	-0.38127
0.17503	1.79321	0.22898

Table 5: Internal input values for polynomial fitting

4.4.3 Fitting procedure

The polynomial formulation in a single line is (abbreviated* p, y and t):

p = b40***y**^4 + b31***y**^3***t** + b30***y**^3 + b22***y**^2***t**^2 + b21***y**^2***t** + b20***y**^2 + b12***y****t**^2 + b11***y****t** + b10***y** + b02***t**^2 + b01***t** + b00.

Here (*) for the sake of abbreviation we have used **p** for **P_poly**, **t** for **T_poly** and **y** for **C_poly**.

b40 is equal to 0 in this polynomial, so this parameter is not taken into account when fitting. However, it must still be written into the OTP (see section 5).

At this point it is convenient to switch to matrix-vector multiplication notation. The line above is found back as:

$$\{p\} = [M]\ \{b\},$$

where 15 pressures are in vector {p}, the 11 coefficients (b31,b30, etc.) are in vector {b} and the columns of matrix [M] hold elementwise products of the 15 C_poly and T_poly numbers.

In order to find vector {b}, the matrix equation has to be used in an inverted sense, according to

$$\{b\} = (M^T M)^{-1} M^T \{p\},$$

using the matrix transpose (T) and inverse (-1) operations. This procedure is commonly known as the Moore-Penrose inverse, or pseudo-inverse. It can be proven to give a least-squares solution for {b}. The pseudo-inverse is valid for overdetermined systems of equations: since we have 15 observations and seek 12 coefficients, our matrix M is non-square and simple inversion would require a square matrix. Using the pseudo-inverse notation, the calculation of coefficients b is performed by matrix-vector multiplications using M and {p}.
For implementation in code, the below lines describe one method of constructing matrix M. Along the orders of the variables and following the naming of coefficients b in the polynomial above, we define

orderY = [3 3 2 2 2 1 1 1 0 0 0], and
orderT = [1 0 2 1 0 2 1 0 2 1 0].

Then the matrix M can be constructed by adding columns in a loop: (MATLAB syntaxis used: ".^" for elementwise power, ".*" for elementwise multiplication)

```
M=[]; % Define empty matrix.
for q=1:numel(orderY)
        M=[M (C_POLY.^orderY(q)).*(T_POLY.^orderT(q))]; % Add columns to the right.
end
```

So that 11 times a column of 15 elements is added to the right of matrix M. The last column refers to orders 0 for both Y and T, so the last column of M holds only 1's.

With M known and vector p available, the coefficients are calculated with the pseudo-inverse. Again using MATLAB syntaxis for illustration, we calculate

b = pinv(M) * p.

For digital storage the coefficient vector b is **multiplied by 2^18**.

Rounding the result found for b gives the 11 numbers in the Table 6 below in column "coef_poly", with a 0 in the position b40:

	coef_poly	ROM	OTPcoef	bits	newout
b40	0	0	-86	9	426
b31	16	-75	91	10	91
b30	4776	3657	1119	15	1119
b22	-16	18	-34	7	94
b21	180	-181	361	11	361
b20	-17192	-4508	-12684	18	249460
b12	-172	-129	129	9	129
b11	2916	3025	-109	13	8083
b10	-221250	-207093	-14157	19	510131
b02	304	135	169	11	169
b01	-3764	-2246	-1518	14	14866
b00	104198	77723	26475	20	26475

Table 6: Extracted calibration coefficients and resulting linearization parameters to be written in OTP taking into account preset correction values in ROM

The chip holds preset **ROM** (see Table) values for the 12 coefficients. The trimming procedure then stores values in the one-time-programmable (OTP) memory cells such that for each coefficient value = **ROM+OTP**. Therefore the coefficient written to the OTP ("OTPcoef") needs to be calculated by OTPcoef = coef_poly-ROM.

4.4.3.1 Fitting procedure with less degrees of freedom

If it is required to include less degrees of freedom in the fit, it is recommended to leave out higher order coefficients for example coefficients b40, b31 & b22 are set to zero. Please note that these three coefficients only marginally adjust the pressure linearization. With a reduced number of P,T calibration points, the polynomial fit will generally give a more stable result if the fitting is limited to 9 coefficients.

Below you can find a MATLAB script that solves for 9 coefficients:

```
orderY = [  3  2 2 1 1 1 0 0 0];
orderT = [  0  1 0 2 1 0 2 1 0];

M=[]; % Define empty matrix.
for q=1:numel(orderY)
        M=[M (C_POLY.^orderY(q)).*(T_POLY.^orderT(q))]; % Add columns to the right.
end
b = pinv(M) * p.
```

4.4.4 Parameter b40 and b12

Note that the parameters b40 and b12 need to have a special treatment for storage. During the development of the chip, the available range for b12 was found to be insufficient and space used for b40 (which is zero anyway) is now used to enlarge the range of b12.

Since for every coefficient there is a ROM part and an OTP part, this means chip-internal polynomial calculation uses:

B12_calculation = [B12_rom + B12_OTP] + 2 * [B40_rom + B40_OTP]

Both original coefficients had 9 bits available. The ROM offsets are:

B40_rom = 0
B12_rom = -129

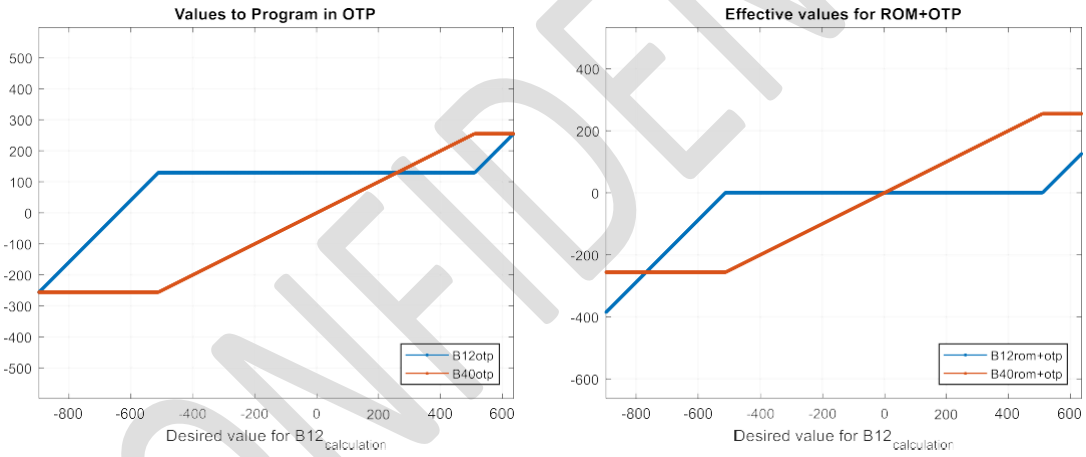

Figure 7: The extended value range for B12 is obtained by using B12+2*B40

In the APPENDIX, the MATLAB implementation is given on one solution to distribute a desired value for B_12 calculation over the two registers B12_OTP and B40_OTP. This is also illustrated in the Table 6 above.

For the blue lines: the central horizontal parts are alternating values to account for odd /even numbers. On the left these values are 129 and 130, on the right these values are 0 and 1.

Note: in order to write the desired value for B12 and B40 into the OTP, conversion to two's-complement will be required. Both have 9 bits reserved, so that values from 0—511 can be stored, of which the second half denote negative values. (see also section 4.4.5)

If we then account for the ROM values and perform the **mentioned distribution of b12_calc** over b12 and b40, we get the values in the column **OTPcoef**.

Again, these numbers are only for this specific example. These 12 numbers represent the least-squares fit so that over the (P,T)-space sampled by the 15 points listed in
Table 2 the pressure polynomial produces a (scaled) value for pressure P_POLY, based on the measured temperature and capacitance, via the inputs T_POLY and C_POLY.

4.4.5 Conversion to two complements

As a last step, conversion into two's- complement is performed. The reserved bits for each coefficient are listed and all negative numbers in **OTPcoef** will appear as large positive numbers in **newout**.

Together with the A and alpha for the temperature sensor, we can add one line to a file called "mergedout.txt" which then looks like this:

UID	alpha_OTP	b10	b02	A_OTP	b01	b40	b31	b00	b30	b22	b21	b20	b12	b11
0x21c0f135	161	904	426	91	1119	94	361	249460	130	8083	510131	169	14866	26475

A final script (See Appendix) will perform the bit-mapping to the P2RAM. The code that represents the mapping table is included in the Appendix. The binary values to be written in addresses 34 till 4F then are:

'01101001'

'00010010'

'10101001'

'10110011'

'10101010'

'10010011'

'10000010'

'01011111'

'01101011'

'01110100'

'10001000'

'10100001'

'00000011'

'01011110'

'01100111'

'00010000'

'11001110'

'00000011'

'10000100'

'01011011'

'00111010'

'00000000'

'00011111'

'00000000'

'00000000'

'00000000'

'10000000'

'01111100'

With the following MATLAB lines these binary numbers can be converted to hexadecimal numbers:

```
>> d=bin2dec('01111100')
d = 124

>> h=dec2hex(d)
h ='7C'
```

Note that some registers in the P2RAM already contain some electrical trimming information. By writing new values to the OTP, only additional 1's will be stored by burning fuses. Overwriting with 0's has no effect.

These are again the resulting values to be written into the P2RAM registers: (both the addresses as well as the values to write are in HEX format)

34	69	39	93	3E	88
35	12	3A	82	3F	A1
36	A9	3B	5F	40	03
37	B3	3C	6B	41	5E

38	AA	3D	74	42	67
43	10	44	CE	45	03
46	84	47	5B	48	3A
49	00	4A	1F	4B	00
4C	00	4D	00	4E	80
4F	7C				

In the section 5 the methodology is described how to store the found parameters in the memory.

4.5 Calibration validation example

As an intermediate form of validation – for illustration purposes – we can measure more points in (P,T) space for this device and then take the output capacitance values, perform the pressure polynomial calculation (in the forward sense, with now known parameters b40, b31, etc.) and determine what is the residual of found Pressure minus recorded reference pressure.

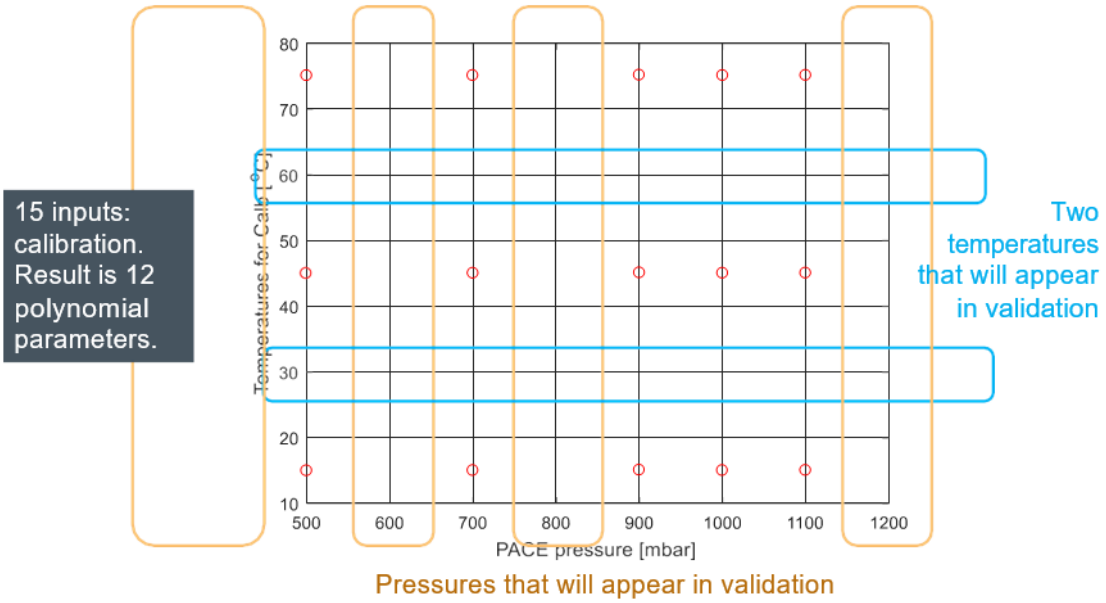

Figure 8: The calibration grid of points in (P,T) space is complemented by additional pressures and temperatures.

We extend the sampling space. Pressures in the range 300 to 1200 hPa are included, as well as two additional temperatures. A completely filled grid of (P,T) points is set, equilibrated, and measured. Based on the previously found 12 parameters, we calculate the residuals and plot these against temperatures and against pressures (see Figure 9).

The three graphs in Figure 9 show three times the same residuals, three times on the vertical axis. Horizontally, the plots are (in order) versus set pressure, set temperature and finally versus time. The provided 15 points in (P,T)-space produce errors very close to zero. The overall performance can be expressed by the infinity-norm of all residuals. This is the maximum residual in the absolute sense.

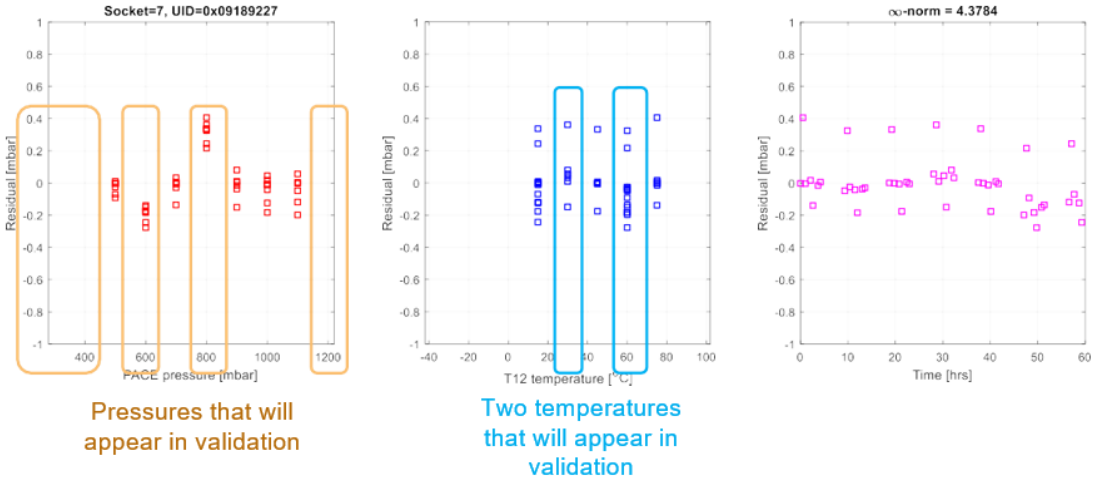

Figure 9: Pressure residuals using the using the calibration constants after evaluating the polynomial output for the chip

In these graphs (limited to +1 and -1 hPa) the maximum residual seems to be around 0.4 hPa. However, as can be seen in the title of the third sub-figure, the norm is indicated to be 4.38 hPa. Zooming out we learn that these errors can be found at the edges of the validation (P,T)-space (see Figure). The behaviour of the residual versus temperature (central plot) seems to be very flat at all pressure levels. The error versus pressure shows a phenomenon typical for polynomial fitting: large errors for extrapolation. For a certain application, the user might optimize here: the calibration regime was chosen equal to the high-performance regime for this example.

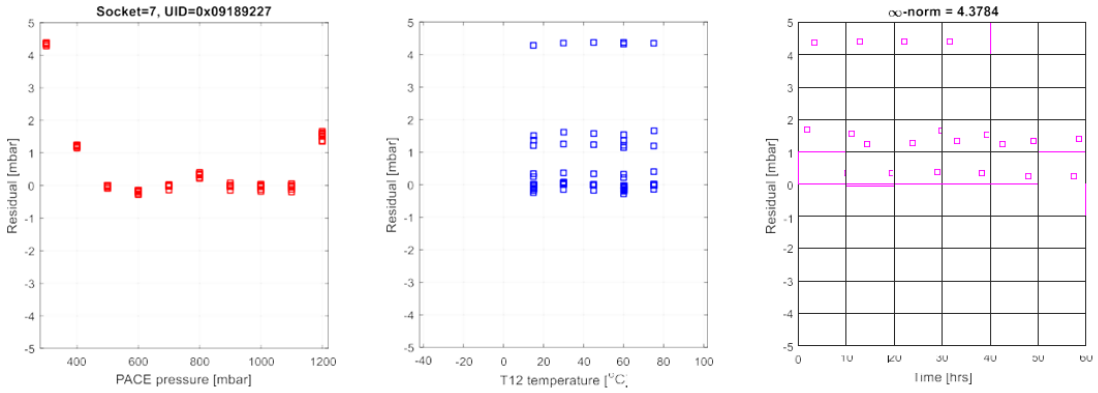

Figure 10: Evaluation of Perror in extended P&T range: Zooming out reveals that in this example the pressure setpoint 300 hPa produces the largest infinity-norm residual of 4 hPa

In these graphs (limited to +1 and -1 hPa) the maximum residual seems to be around 0.4 hPa. However, as can be seen in the title of the third sub-figure, the norm is indicated to be 4.38 hPa. Zooming out we learn that these errors can be found at the edges of the validation (P,T)-space (see Figure). The behaviour of the residual versus temperature (central plot) seems to be very flat at all pressure levels. The error versus pressure shows a phenomenon typical for polynomial fitting:

large errors for extrapolation. For a certain application, the user might optimize here: the calibration regime was chosen equal to the high-performance regime for this example.

5 Trimming: Writing to OTP and burning fuses

5.1 P2RAM (OTP) map

The chip uses two P2RAM OTP blocks for trimming. Both P2RAMs are integrated in the chip digital core. Their special registers will be accessed via the device I2C interface. P2RAM blocks have 4 words.

5.1.1 P2RAM1

Addr	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
30	UID0[7:0]							
31	UID1[7:0]							
32	UID2[7:0]							
33	UID3[7:0]							
34	b21[7:0]							
35	b01[7:0]							
36	b02[7:0]							
37	b10[7:0]							
38	b40[7:0]							
39	b11[7:0]							
3A	b12[7:0]							
3B	b30[7:0]							
3C	b00[7:0]							
3D	b20[7:0]							
3E	A[7:0]							
3F	alpha[7:0]							

Table 1: Map of PTRAM1

5.1.2 P2RAM2

Addr	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
40	agnd[1:0]				A[13:8]			
41	I2C[0]	b22[6:0]						
42	b00[15:8]							
43	B12[8]	b21[10:8]			b00[19:16]			
44	b20[15:8]							
45	alpha[12:8]					I2C[1]	b20[17:16]	
46	b40[8]	b30[14:8]						
47	b31[7:0]							
48	b31[9:8]				b01[13:8]			
49	cdc_offset[2:0]			Reserved	PARTID[3:0]			
4A	b02[10:8]				b11[12:8]			
4B	osc_slow[7:0]							
4C	cdc_ref[1:0]				osc_fast[5:0]			
4D	ibias[3:0]				cdc_ext	cdc_ref[4:2]		
4E	b10[11:8]					I2C[4:2]		ibias[4]
4F	Reserved	b10[18:12]						

Table 2: Map of PTRAM2

5.2 P2RAM (OTP) functions

5.2.1 P2RAM WRITE

The P2RAMs can be written by:

- Applying an I^2C write command with the data 0xFD at the corresponding P2RAM special register.
- Applying an I^2C write command with the wanted data to the corresponding P2RAM register
- After writing the data in the register, the register changes are applied. The written register data will be lost after power-down the device or supply voltage is removed.

5.2.2 P2RAM READ

The read of the P2RAMs register is done by directly applying an I^2C read command at the P2RAM register address.

5.2.3 P2RAM BURNING

A 10uF capacitor must be connected to SDA during the trimming procedure. The SDA voltage is applied to the P2RAM during the trimming procedure. The SDA voltage must be set at 2.7V during fusing. During the trimming procedure the clock must be provided externally at SCL. The external clock is selected by setting test mode TM30.

Name	Reg. Add. (HEX)	Test mode	Bit	Function and comment
ANA_TM2	53	TM30	0	SEL_EXT_CLOCK: the main clock is coming from SCL

5.3 Burning sequence

It is advisable to keep VDD at 1.8V:

```
0x06 = 0x80     // high power mode
WAIT 1ms
0x7F = 0x67     // test mode activation
0x50 = 0xFD     // enable P2RAM1 (special register)
0x51 = 0xFD     // enable P2RAM2 (special register)
```

Write P2RAMs content (address from 0x30 to 0x3F for P2RAM1 and 0x40 to 0x4F for P2RAM2)

```
0x50 = 0x08     // P2RAM1 burn condition (special register)
0x51 = 0x08     // P2RAM2 burn condition (special register)
```

Execute a custom I^2C command as follows:

- Send Start
- Send slave address + write bit (0x40)
- Device acknowledges write

- Send address of first byte to fuse (e.g. 0x30 for P2RAM1)
- Device acknowledges write
- Send Start
- Send slave address + read bit (0x41)
- Device acknowledges write
- Keep SCL low while switching SDA to 2.7V; no time constraint for this operation
- Start pulsing SCL at 1.25KHz for 9*N - 1 times. N is the number of bytes to be fused (N=32 to fuse both P2RAM1 and P2RAM2).
- Set SDA=0; no time constraint for this operation
- Give a pulse to SCL to signal the end of the burning sequence
- Send Stop or power cycle the device
- Send soft reset (0x6=0x8) if there was no power cycle

The diagrams shown in Figure 11 below further exemplify custom command in case of complete fusing of P2RAM1 & P2RAM2:

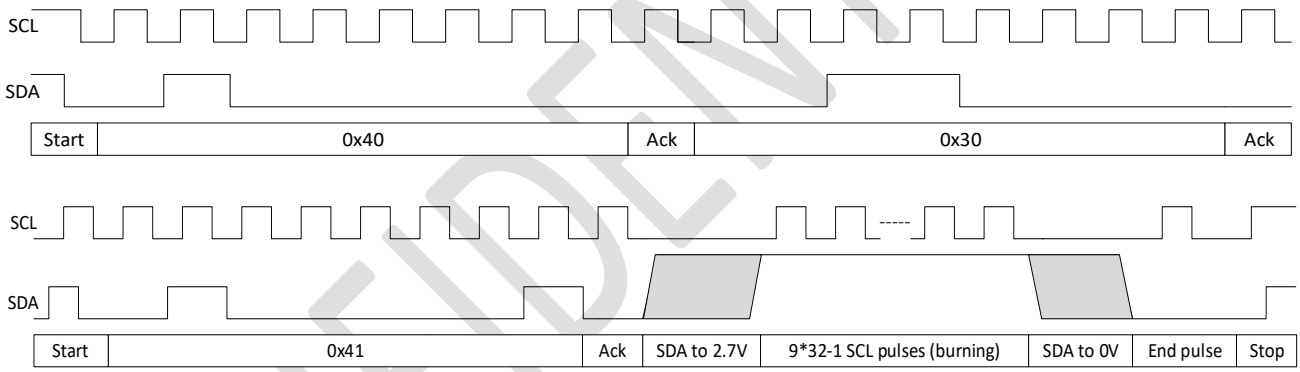

Figure 11 Burning sequence

5.4 Fuse board design

Figure 12 below is a schematic diagram showing the circuitry connected to SDA: normally it is pulled-up to VDD; at the right moment, as described above, the switch is turned-on and SDA is connected to a fixed 2.7V power supply.

Figure 13 below is a schematic for a parallel fuse board design is shown, it is suggested to use one relay per device. The relay is an electro-mechanical switch having a low impedance. We suggest to implement 100nF decoupling for each DUT. Solid state relays may add a significant voltage drop during fusing, therefore we recommend mechanical relays; peak currents on SDA may reach 40mA.

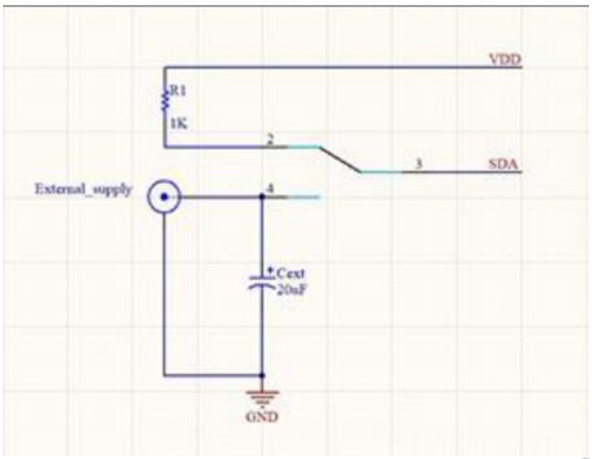

Figure 12 Fuse board design single device

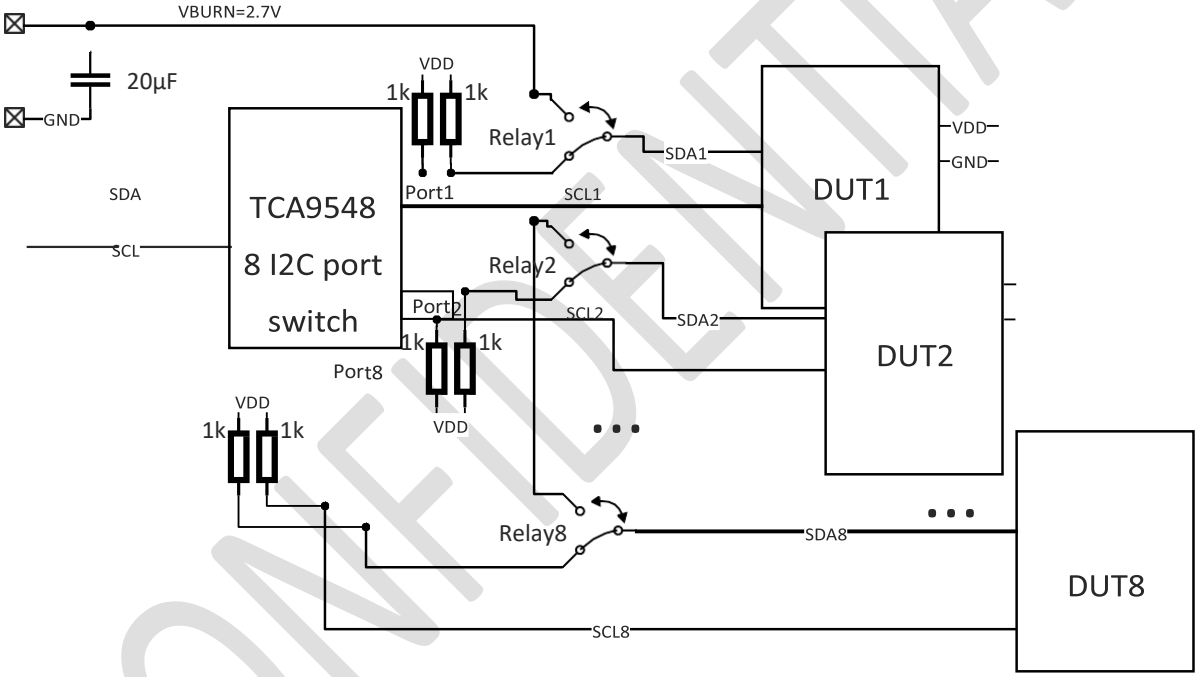

Figure 13 Fuse board design for parallel fusing

6 Appendix

6.1 General information

6.2 Register information

This section describes the registers of chip. More detailed info can be found in the chip datasheet.

Address	Name	Type	Access in LP mode	Default	Description
0x00	PART_ID0	R	Y	0x2X	Part ID [7:0]
0x01	PART_ID1	R	Y	0x03	Part ID [15:8]
0x02	UID0	R	N		Unit ID [7:0]
0x03	UID1	R	N		Unit ID [15:8]
0x04	UID2	R	N		Unit ID [23:16]
0x05	UID3	R	N		Unit ID [31:24]
0x06	MODE_CFG	R/W	Y	0x00	Device configuration
0x07	MEAS_CFG	R/W	N	0x08	Measurement configuration, conversion time
0x08	STBY_CFG	R/W	Y (W)[2]	0x00	Standby time configuration
0x09	OVS_CFG	R/W	N	0x00	Oversampling setting for pres- sure and temperature
0x0A	MAVG_CFG	R/W	N	0x00	Moving average configuration
0x0B	INTF_CFG	R/W	Y (W)[2]	0x00	Interface configuration
0x0C	INT_CFG	R/W	Y (W)[2]	0x7F	Interrupt mask configuration
0x0D	PRESS_LO_XL	R/W	N	0x00	Low pressure threshold [7:0]
0x0E	PRESS_LO_L	R/W	N	0x00	Low pressure threshold [15:8]
0x0F	PRESS_LO_H	R/W	N	0x00	Low pressure threshold [23:16]
0x10	PRESS_HI_XL	R/W	N	0xFF	High pressure threshold [7:0]
0x11	PRESS_HI_L	R/W	N	0xFF	High pressure threshold [15:8]
0x12	PRESS_HI_H	R/W	N	0xFF	High pressure threshold [23:16]
0x13	FIFO_CFG	R/W	N	0x00	FIFO configuration
0x14	DATA_STAT	R	Y	0x00	Measurement data status
0x15	FIFO_STAT	R	N	0x02	FIFO status

0x16	INT_STAT	R	Y	0x08	Interrupt status
0x17	PRESS_OUT_XL	R	Y[3]	0x00	Pressure value [7:0]
0x18	PRESS_OUT_L	R	Y[3]	0x00	Pressure value [15:8]
0x19	PRESS_OUT_H	R	Y[3]	0x00	Pressure value [23:16]
0x1A	TEMP_OUT_L	R	Y	0x00	Temperature value [7:0]
0x1B	TEMP_OUT_H	R	Y	0x00	Temperature value [15:8]
0x27	PRESS_OUT_F_XL	R	Y[3]	0x00	Pressure value [7:0]
0x28	PRESS_OUT_F_L	R	Y[3]	0x00	Pressure value [15:8]
0x29	PRESS_OUT_F_H	R	Y[3]	0x00	Pressure value [23:16]

Table 3: Register overview

[2] Write access only
[3] Only the latest value; see description for detailed info.

6.3 Temperature calibration code

Some MATLAB functions and code used for temperature calibration can be found below.

```
function [ALPHAvalue,Avalue]=newfinding(ensTs,T12s)
temp_out=((ensTs+273.15)*128); % insert temperatures as measured on ens when no fusing was done (uncalibrated)
[X]=Temp2Raw(temp_out, 0, 0); % Produce raw values
% find proper alpha and a
lambda0=[150 900];
options = optimset('TolX',1e-19,'TolFun',1e-19,'MaxIter',29199,'MaxFunE- vals',5930,'Display','on');
[lam,fval]=fminsearch(@(lambda)findvals(lambda, X, T12s),lambda0,options);
Avalue=lam(2);
ALPHAvalue=lam(1);


function [err]=findvals(lam, X, T12s) atrim=lam(2);
alphatrim=lam(1);
[T]=Raw2Temp(X, alphatrim, atrim); % From raw values to temperatures, WITH the proper A and Alpha
EnsAdjusted=(T-34963.2)/128; % for the true Kelvin readings
err=sum((EnsAdjusted-T12s).^2);

function [X]=Temp2Raw(temp_out, alpha_trim, A_trim)
alpha = alpha_trim * 512 + 8388608;
A = A_trim * 8192;
```

```
X4 = ((temp_out)+2029)*4;
X3 = (X4 ./ (1 + A / 2^30));
X31 = (X3 ./ (1 + 23068672 / 2^27));
X2=X31*(2^25)/alpha; X1 = (2^43 ./ X2);
X = ((X1 - alpha) / 32);

function [T]=Raw2Temp(raw, alpha_trim, A_trim) alpha = alpha_trim * 512 + 8388608;
A = A_trim * 8192;
X1 = raw * 32 + alpha; X2 = (2^43 ./ X1);
X31=X2*alpha/(2^25);
X3 = X31+((X31 * 23068672) / 2^27); X4 = X3 + (X3 * A / 2^30);
T = X4/4-2029;

function [b12progs, b40progs]=b12mapper(vals)
% values in 'vals' are allowed to be -897 < val < 636. Those are values for
% true coefficients, so ROM+OTP b12progs=0*vals; b40progs=0*vals;
for q=1:numel(vals) val=vals(q);
% determine if val is within the range of 2*b40
        if (val>-513)&(val<510)
                b40prog=floor(val/2);
                if b40prog==val/2 b12prog=0; % ignore ROM here else
                        b12prog=1; % odd/even
                end
        elseif (val<-512)
                b40prog=-256;
                b12prog=val-2*b40prog;
        else
                b40prog=255;
                b12prog=val-2*b40prog;
        end

        b12progs(q)=b12prog;
        b40progs(q)=b40prog;
end
```

6.4 Mapping bits to P2RAM addresses

MATLAB code for mapping the bits of polynomial parameters to the corresponding P2RAM addresses can be found below:

```
ab=[161 904 426 91 1119 94 361 249460 130 8083 510131 169 14866 26475]; % 2+12 coefficients, example

nbs=[13 14 9    10 15 7 11 18 9 13 19 11 14 20]; % number-of-bits
% alpha_OTP A_OTP     b40 b31 b30 b22 b21 b20 b12 b11 b10 b02 b01 b00
%       1       2       3       4       5       6       7       8       9       10      11      12
        13      14
bbs={};

for q=1:length(ab) % for each of the 14 parameters
        bij=ab(q);

        % convert to binary, taking the number of reserved bits per parameter
        bbs{q}=dec2bin(bij,nbs(q));
end

% generate bytes: for 28 addresses (starting 34 and ending 4f)
newbytes='22222222'; % initialize this variable as a string of 8 characters and do not use 0 or 1 in that string.

newbytes(1,:)=[bbs{7}(end-7:end)]; % 0x34, to be added, b21
newbytes(2,:)=[bbs{13}(end-7:end) ]; % 0x35, b01
newbytes(3,:)=[bbs{12}(end-7:end) ]; % 0x36, b02
newbytes(4,:)=[bbs{11}(end-7:end) ]; % 0x37, b10
newbytes(5,:)=[bbs{3}(end-7:end) ]; % 0x38, b40
newbytes(6,:)=[bbs{10}(end-7:end) ]; % 0x39, b11
newbytes(7,:)=[bbs{9}(end-7:end) ]; % 0x3A, b12
newbytes(8,:)=[bbs{5}(end-7:end) ]; % 0x3B, b30

newbytes(9,:)=[bbs{14}(end-7:end) ]; % 0x3C, b00
newbytes(10,:)=[bbs{8}(end-7:end) ]; % 0x3D, b20
newbytes(11,:)=[bbs{2}(end-7:end)]; % 0x3E, A
newbytes(12,:)=[bbs{1}(end-7:end) ]; % 0x3F, alpha
newbytes(13,:)=['00' bbs{2}(end-13:end-8) ]; % 0x40, A
newbytes(14,:)=['0' bbs{6}(end-6:end)]; % 0x41, b22
newbytes(15,:)=[bbs{14}(end-15:end-8) ]; % 0x42, b00
newbytes(16,:)=[bbs{9}(end-8)  bbs{7}(end-10:end-8)  bbs{14}(end-19:end-16)]; % 0x43, b12, b21, b00
```

```
newbytes(17,:)=[bbs{8}(end-15:end-8)]; % 0x44, b20
newbytes(18,:)=[bbs{1}(end-12:end-8) '0' bbs{8}(end-17:end-16) ]; % 0x45, alpha, b20
newbytes(19,:)=[bbs{3}(end-8) bbs{5}(end-14:end-8)]; % 0x46, b40, b30
newbytes(20,:)=[bbs{4}(end-7:end-0)]; % 0x47, b31

newbytes(21,:)=[bbs{4}(end-9:end-8)  bbs{13}(end-13:end-8)];  % 0x48, b31, b01
newbytes(22,:)=['00000000']; % 0x49, NOTHING to FUSE: cdc_offset
newbytes(23,:)=[bbs{12}(end-10:end-8)  bbs{10}(end-12:end-8)];  % 0x4A, b02 and b11
newbytes(24,:)=['00000000']; % 0x4B, osc_slow: do not overwrite

newbytes(25,:)=['00000000']; % 0x4C=cdc_ref + osc_fast
newbytes(26,:)=['00000000']; % 0x4D=ibias + cdc_ref
newbytes(27,:)=[bbs{11}(end-11:end-8) '0000']; % 0x4E
newbytes(28,:)=['0' bbs{11}(end-18:end-12)]; % 0x4F, b10
```